

Making Sense of Declarative Process Models: Common Strategies and Typical Pitfalls^{*}

Cornelia Haisjackl¹, Stefan Zugal¹, Pnina Soffer², Irit Hadar²,
Manfred Reichert³, Jakob Pinggera¹, and Barbara Weber¹

¹ University of Innsbruck, Austria

{cornelia.haisjackl, stefan.zugal, jakob.pinggera, barbara.weber}@uibk.ac.at

² University of Haifa, Israel

{spnina, hadari}@is.haifa.ac.il

³ University of Ulm, Germany

manfred.reichert@uni-ulm.de

Abstract. Declarative approaches to process modeling are regarded as well suited for highly volatile environments as they provide a high degree of flexibility. However, problems in understanding and maintaining declarative business process models impede often their usage. In particular, how declarative models are understood has not been investigated yet. This paper takes a first step toward addressing this question and reports on an exploratory study investigating how analysts make sense of declarative process models. We have handed out real-world declarative process models to subjects and asked them to describe the illustrated process. Our qualitative analysis shows that subjects tried to describe the processes in a *sequential way* although the models represent circumstantial information, namely, conditions that produce an outcome, rather than a sequence of activities. Finally, we observed difficulties with single building blocks and combinations of relations between activities.

Key words: Declarative Process Models, Empirical Research, Understandability.

1 Introduction

Regarding the analysis and design of information systems, conceptual modeling has proven to foster understanding and communication [1]. For example, *business process models* (*process models* for short) have been employed in the context of process-aware information systems, service-oriented architectures, and web services [2]. Recently, *declarative* process models have gained attention due to their flexibility with respect to modeling and execution of processes [3]. While technical issues of declarative process modeling, such as formalization of semantics [4], maintainability [5], verification [6], and execution [7] are well understood, understandability issues of declarative models have not been investigated in detail yet. In particular, it has been argued that understandability may be hampered by

^{*} This research is supported by Austrian Science Fund (FWF): P23699-N23 and the BIT fellowship program. The final publication is available at Springer via http://dx.doi.org/10.1007/978-3-642-38484-4_2

lack of computational offloading [8] or hidden dependencies [9]. Put differently, it is not entirely clear whether the full potential of declarative modeling can be exploited or whether understandability issues will interfere.

We approach these issues by studying the sense-making of declarative process models through the lens of an empirical investigation. In particular, we handed out declarative process models to subjects and asked them to describe the illustrated process. In addition, we asked them to voice their thoughts while describing the process, i.e., we applied *think-aloud techniques* [10] to get insights into the subject’s reasoning processes. Since we were interested in how different structures of the process representation would influence process model understandability, we maintained another variant of each model describing the same process, but making use of modularization, i.e., sub-processes. The contribution of this work is twofold. On one hand, we provide insights into how subjects make sense of declarative process models, e.g., we analyze strategies how to read declarative process models. On the other, we consider characteristic problems that occur when scanning declarative process models. Our contribution aims at guiding the future development of supporting tools for system analysts, as well as pointing out typical pitfalls to teachers and educators of analysts.

The exploratory study reported in this paper is part of a larger investigation on declarative process models. While our previous work focused on quantitative results, this paper deals with qualitative data solely.² Sect. 2 gives background information. Sect. 3 describes the setup of the exploratory study, whereas Sect. 4 deals with its execution. Sect. 5 presents the results of the exploratory study and Sect. 6 a corresponding discussion. Related work is presented in Sect. 7. Finally, Sect. 8 concludes the paper.

2 Background: Declarative Process Models

There has been a long tradition of modeling business processes in an imperative way. Process modeling languages supporting this paradigm, like BPMN and EPC, are widely used. Recently, *declarative approaches* have received increasing interest, as they suggest a fundamentally different way of describing business processes [6]. While imperative models specify exactly *how* things must be done, declarative approaches focus on the logic that governs the interplay of process actions by describing *activities* that may be performed, as well as *constraints* prohibiting undesired behavior. Constraints found in literature may be divided into existence constraints, relation constraints, and negation constraints [11]. *Existence constraints* specify how often an activity must be executed for one particular process instance. In turn, *relation constraints* restrict the ordering of activities by imposing respective restrictions. Finally, *negation constraints* define negative relations between activities. Table 1 shows examples for each category, an overview of all constraints can be found in [11].

² The exploratory study’s material can be downloaded from:
<http://bpm.q-e.at/experiment/HierarchyDeclarative>

| Group | Constraint | Definition |
|-----------|----------------------|---|
| existence | exactly(a,n) | activity <i>a</i> must occur exactly <i>n</i> times |
| | existence(a,n) | <i>a</i> must occur at least <i>n</i> times |
| | init(a) | <i>a</i> must be the first executed activity in every trace |
| | last(a) | <i>a</i> must be the last executed activity in every trace |
| relation | precedence(a,b) | activity <i>b</i> must be preceded by activity <i>a</i> (but not necessarily directly preceded) |
| | response(a,b) | if <i>a</i> is executed, <i>b</i> must be executed afterwards (but not necessarily directly afterwards) |
| | chain_response(a,b) | if <i>a</i> is executed, <i>b</i> is executed directly afterwards |
| | coexistence(a,b) | if <i>a</i> is executed, <i>b</i> must be executed and vice-versa |
| negation | neg_response(a,b) | if <i>a</i> is executed, <i>b</i> must not be executed afterwards |
| | neg_coexistence(a,b) | <i>a</i> and <i>b</i> cannot co-occur in any trace |

Table 1. Definition of constraints

An example of a declarative process model *S* specified with ConDec [6] is depicted in Fig. 1. The model consists of six distinct activities A, B, C, D, E, and F. In addition, it comprises three constraints. The *neg_coexistence* constraint, i.e., C1, forbids that A and B co-occur in the same trace. In turn, the *response* constraint, i.e., C2, requires that every execution of C must be followed by one of F before the process instance may complete. Finally, the *exactly* constraint, i.e., C3, states that F must be executed exactly once per process instance. While instances with traces $\sigma_1 = \langle A, A, D, E, A, F \rangle$, $\sigma_2 = \langle B, C, F, E, B \rangle$, and $\sigma_3 = \langle B, E, F \rangle$ satisfy all the constraints, $\sigma_4 = \langle A, F, C, E, A \rangle$ violates C2, $\sigma_5 = \langle B, D, F, C, F \rangle$ violates C3, and $\sigma_6 = \langle A, D, B, F, E \rangle$ violates C1. $\sigma_5 = \langle B, D, F, C, F \rangle$ highlights a *hidden dependency* between C and F. The combination of the *exactly* constraint, i.e., C3, and the *response* constraint, i.e., C2, adds an implicit constraint that does not exist when looking at the constraints in isolation. This hidden dependency prohibits that F is executed before C, assuming that C is executed at all.

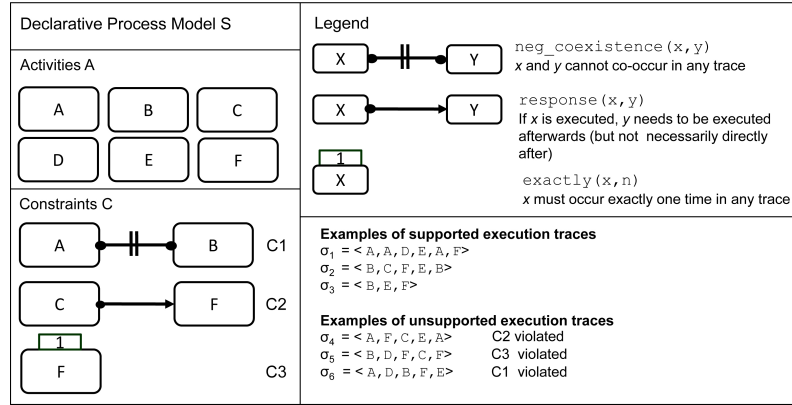


Fig. 1. Example of a declarative process model

Hierarchy in Declarative Process Models. Using modularization to hierarchically structure information has been identified as a viable approach to deal with complexity for decades [12]. Taking a look at declarative process models with hierarchy in general, a sub-process may be introduced in a process model via a *complex activity*, referring to a process model. When the complex activity is executed, the referred process model, i.e., the sub-process, is instantiated (see [13] for details). Fig. 2a) shows a hierarchical model, complex activity *B* refers to a sub-process that contains activities *C* and *D*. Fig. 2b) shows the corresponding flat process model. Even though Fig. 2a) and Fig. 2b) are semantically equivalent, they differ in the number of activities and constraints.

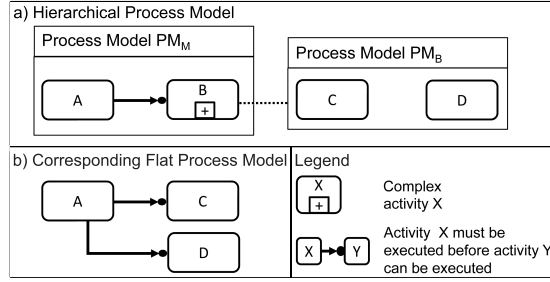


Fig. 2. Example of a process model with and without hierarchy

3 Defining and Planning the Exploratory Study

In order to investigate how subjects make sense of declarative process models we conduct an exploratory study. In particular, we are interested in common strategies and typical pitfalls occurring during this sense-making process. Since there has been no considerable research on understandability issues of declarative process models, and hence no theories exist we can base our investigation on, we address the topic in an exploratory manner using a qualitative research approach [14]. In particular, we use the think-aloud method, i.e., we ask participating subjects to voice their thoughts, allowing for a detailed analysis of their reasoning process [10]. Then, we turn to grounded theory [15], an analysis approach for identifying recurring aspects and grouping them to categories. These categories are validated and refined throughout the analysis process. First of all, we describe setup and planning of the exploratory study.

Subjects. In order to ensure that obtained results are not influenced by unfamiliarity with declarative process modeling, subjects need to be sufficiently trained. Even though we do not require experts, subjects should have at least a moderate understanding of declarative processes' principles.

Objects. The process models used in the study originate from a case study [16] and describe real-world business processes. From a set of 24 process models collected in this case study, 4 models were chosen as basic *objects* for the exploratory

study. This was accomplished in a way ensuring that the numbers of activities and constraints vary. To make the models amenable for this study, they underwent the following procedure. First, the models were translated to English (the case study was conducted in German) since all exercises were done in English (four subjects did not speak German). Second, since the models collected during the modeling sessions had not gone through quality assessment, they were scanned for errors and corrected accordingly. Third, since we were interested in how different structures of the process representation would influence the process models' understandability, we created a second variant of each process describing the same process, but making use of sub-processes. Consequently, we have two variants of each process model: a flat and a hierarchical one.

| | Type | Proc. 1 | Proc. 2 | Proc. 3 | Proc. 4 |
|---------------|-----------|----------------------|----------|--------------------|---------------------|
| Activities | flat | 11 | 8 | 23 | 23 |
| | hierarchy | 13 | 9 | 26 | 26 |
| Constraints | flat | 19 | 7 | 30 | 45 |
| | hierarchy | 21 | 9 | 28 | 44 |
| Constr. types | | 8 | 4 | 7 | 5 |
| Sub-processes | hierarchy | 2 | 1 | 3 | 2 |
| Components | | 2 | 5 | 2 | 2 |
| Domain | | Software development | Teaching | Electronic company | Buying an apartment |

Table 2. Characteristics of the process models used in this study

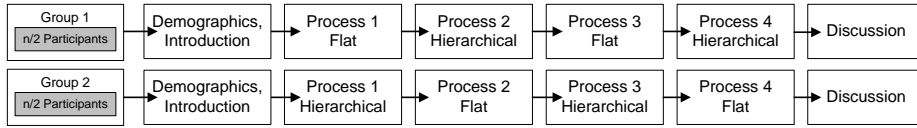
Table 2 summarizes the characteristics of the process models. The latter comprise between 8 and 26 activities, and between 7 and 45 constraints. The differences in the number of activities between flat and hierarchical models are caused by the complex activities representing sub-processes in the hierarchical models (cf. Sect. 2). Similarly, constraints had to be added or removed to preserve the behavior when creating a hierarchical model. Process models vary regarding the degree of interconnectivity of constraints, i.e., models consist of two to five components (cf. Table 2). A component is defined as a part of the model where any two activities are connected by constraints, and not connected to any other activity in the model. The process models are based on four different domains describing bug fixing in a software company, a teacher's preparations prior to teaching, a worker's duties at an electronic company, and buying and renovating an apartment (cf. Table 2). The process models contain constraints of all three types, i.e., existence, relation, and negation constraints, except the second process model (no negation constraints). Table 3 provides additional information on the constraint types included in each process model.

Design. Fig. 3 shows the overall design of the exploratory study: First, subjects are *randomly* assigned to two groups of similar size. Regardless of the group assignment, demographical data is collected and subjects obtain introductory assignments. To support subjects in their task, cheat sheets briefly summarizing the constraints' semantics, are provided, which can be used throughout the study. Introductory tasks allow subjects to familiarize themselves with the type

| Group | Constraint | flat | | | | hierarchical | | | |
|-----------|-----------------------|------|-----|-----|-----|--------------|-----|-----|-----|
| | | P 1 | P 2 | P 3 | P 4 | P 1 | P 2 | P 3 | P 4 |
| existence | existence constraints | 5 | 2 | 1 | 10 | 7 | 4 | 1 | 13 |
| | init | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |
| | last | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| relation | precedence | 4 | 3 | 18 | 20 | 4 | 3 | 18 | 20 |
| | response | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| | succession | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 4 |
| | coexistence | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| | chained precedence | 2 | 0 | 0 | 0 | 2 | 0 | 0 | 0 |
| | chained response | 3 | 0 | 1 | 0 | 3 | 0 | 1 | 0 |
| | chained succession | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| | | | | | | | | | |
| negation | negation response | 2 | 0 | 0 | 10 | 2 | 0 | 0 | 6 |
| | mutual exclusion | 1 | 0 | 7 | 0 | 1 | 0 | 5 | 0 |

Table 3. Constraints of the process models used in this study

of tasks to be performed—potential problems can therefore be resolved at this stage without influencing actual data collection. After this familiarization phase, subjects are confronted with the actual tasks. Each subject works on two flat process models and two hierarchical ones. Group 1 starts with the flat representation of process model 1, while Group 2 works on the hierarchical representation of the same model. Subjects are confronted with hierarchical and flat models in an alternating manner. For each model, the subject is asked to “*explain roughly what the process describes.*” The exploratory study is concluded by a discussion with the subject to help reflecting on the study and providing us with feedback.

**Fig. 3.** Design of the exploratory study

Instrumentation. For each model, subjects received separate paper sheets showing the process models, allowing them to use a pencil for highlighting or taking notes, and juxtaposing the process models as desired. No written answers were required, only free talking. Audio and video recording are used as it has proven being useful for resolving unclear situations in think-aloud protocols [17].

4 Performing the Exploratory Study

Execution. The study was conducted in July 2012 in two locations. First, seven subjects participated at the University of Ulm, followed by two additional sessions at the University of Innsbruck, i.e., a total of nine subjects participated. To ensure that subjects were sufficiently familiar with declarative process modeling, they were provided with training material. Each session was organized as follows: First, the subject was welcomed and instructed to speak thoughts out loudly.

To allow subjects to concentrate on their tasks, the sessions were performed in a “paper-workflow” manner, i.e., one supervisor was seated left to the subject, a second supervisor to the right. The sheets containing the study’s material were then passed from the left to the subject. As soon as the subject finished the task, the material was passed to the supervisor on the right. Meanwhile, the subject’s actions were audio- and video-recorded to gather any uttered thoughts.

Data Validation. In each session, only a single subject participated, allowing us to ensure that the study setup was obeyed. In addition, we screened whether subjects fitted the targeted profile, i.e., were familiar with process modeling and ConDec [6]. We asked questions regarding familiarity on process modeling, ConDec, and domain knowledge; note that the latter may significantly influence performance [18]. For this, we utilize a 7-point Likert Scale, ranging from “*Strongly agree*” (7) over “*Neutral*” (4) to “*Strongly disagree*” (1). Results are summarized in Table 4. Finally, we assessed the subjects’ professional background: all subjects indicated an academic background, i.e., were either PhD students or postdocs. We conclude that they had a profound background in process modeling (the least experienced subject had 2.5 years of modeling experience) and were moderately familiar with ConDec.

| | Minimum | Maximum | Median |
|--------------------------------------|---------|---------|--------|
| 1) Years of modeling experience | 2.5 | 7 | 5 |
| 2) Models read last year | 10 | 250 | 40 |
| 3) Models created last year | 5 | 100 | 10 |
| 4) Average number of activities | 5 | 50 | 15 |
| 5) Familiarity ConDec | 2 | 6 | 3 |
| 6) Confidence understanding ConDec | 2 | 6 | 4 |
| 7) Confidence creating ConDec | 2 | 6 | 4 |
| 8) Familiarity software development | 4 | 7 | 6 |
| 9) Familiarity teaching | 4 | 7 | 5 |
| 10) Familiarity electronic companies | 1 | 6 | 2 |
| 11) Familiarity buying apartments | 1 | 6 | 4 |

Table 4. Demographics (5–11 based on 7-point Likert Scale)

Data Analysis. Our research focuses on sense-making of declarative process models. On one hand, we investigate strategies applied by subjects in understanding process models, on the other, we explore typical phenomena and pitfalls in this process. For this purpose, data analysis comprised the following stages.

1. Transcription of the subjects’ verbal utterances
2. Creation of graphs describing the order in which subjects mention activities
3. Analysis of transcripts using grounded theory

In (2), for each process model we create a graph representing the order activities were mentioned by the subjects. For this purpose, we utilize the transcripts created in (1), but also video recordings to identify when subjects visited an activity without talking about it. In (3), we apply grounded theory to

the transcripts to explore and understand phenomena appearing when subjects make sense of declarative process models. As a starting point, transcripts are inspected, marking aspects that caused confusion, were misinterpreted or left out. In a second iteration, we revisit the marked areas and search for new aspects. This process of open coding analysis is repeated until no new aspects can be found. Afterwards, we perform axial coding, i.e., we repeatedly group aspects to form high level categories. We count the number of identified markings per category.

5 Findings

Based on the findings of our data analysis, we identified different ways how declarative models are read and interpreted.

5.1 Reading Declarative Business Process Models

When analyzing graphs and transcripts, we observed that subjects consistently adopted similar strategies when reading declarative models. For example, Fig. 4 shows the flat version of the first model and a typical strategy to understand that model. The model consists of two components. The first one contains activities “*receive bug report*” and “*search for bug in archive*”. The second component comprises all other activities. The dotted arrows display how three out of five subjects (Group 1) read the model to understand it.

Regardless of whether sub-processes were present or not, they described the process in the order activities were supposedly executed, i.e., tried to describe the process in a *sequential way*. Hence, as a first step, subjects skimmed over the process model to find an *entry point* where they could start with describing the (main) process: “... *Ok, this is the first activity since it has this init constraint...*” Interestingly, subjects appreciated when a clear starting point for their explanations could be found: “... *it is nice that we have an init activity, so I can start with this...*” Relating to the model depicted in Fig. 4, subjects started with “*receive bug report*” because of the init constraint. Then, they mentioned “*search for bug in archive*”. A declarative process model, however, does not necessarily have a unique entry point, apparently causing confusion: “*Well...gosh...I’ve got no clue where to start in this model...*” The subjects used two different solutions for this kind of situation. Either they looked for a last constraint (“*So, we don’t have init, but we have last...*”) or they assumed the upper left corner of the model to be its entry point (“*Ok...so first of all I have three initial I would say start activities...*”). After having identified an entry point, subjects tried to figure out in which order activities are to be executed: “*After given duties to the apprentices there should come these two tasks...*”

This routine was iterative, i.e., if parts of a model were not connected, subjects applied the same strategy for each component, i.e., they started again at the upper left corner of these components. We observed this behavior independent of the respective process model or subject. Regarding our example (cf. Fig. 4), after describing the first component, subjects took a look at the second one. As there

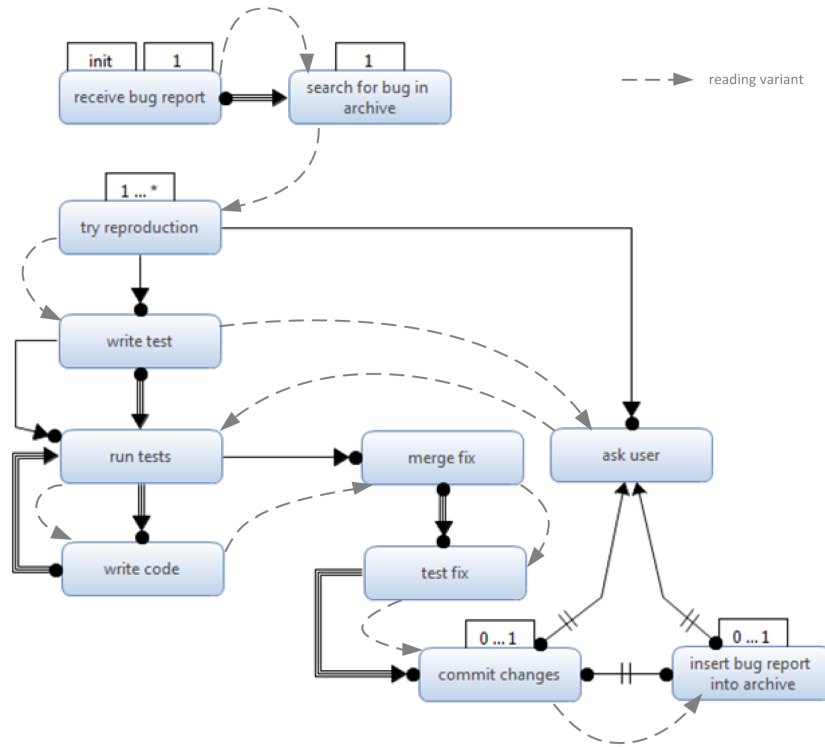


Fig. 4. First process model and a reading variant

was no init constraint, they started in the upper left corner (“try reproduction”) and followed the other activities in a sequential way. Two subjects had problems, since there is no connection between the two parts of the model: “Ah, then there is a different process because these are not connected. . .” Likewise, a subject got irritated with single activities that had no connection to the rest of the model: “...ah this one, there’s no constraint here. You are just trying to confuse me.” Finally, subjects indicated where the process supposedly ends: “...the process ends with the activity give lessons. . .” When there was no last constraint (cf. Fig. 4), subjects stopped describing the process model after having mentioned all activities of all components.

If a model contained sub-processes (cf. Table 2), subjects preferred talking first about the main process in the above specified way before describing the sub-processes. When reading sub-processes the subjects used the same routine as for the main process, except two subjects. One of them described all and the second subject one out of four sub-processes completely *backwards*, i.e., following the semantics of precedence constraints, instead of describing them sequentially.

5.2 Single Building Blocks

Flat Declarative Process Models. In general, when subjects try to make sense of a model, they name activities and their connections. Sometimes, it happened that subjects missed single or small groups of activities. Regarding the first and second flat process model, no activities were left out. Three out of five subjects missed either 2, 4 and 17 activities in the flat version of the third process model (26 activities). The 17 activities were not referred to because a subject did not look at the components of the process model in detail. Four activities were not mentioned in the fourth flat process model: two out of four subjects did not mention one and three activities out of 26 activities. In summary, 27 out of 294 activities were missed in flat process models.

When describing a model sequentially, subjects name activities explicitly and most of the connections, i.e., the constraints, implicitly. However, most subjects did not mention existence constraints. This behavior could not be found for any other constraint. For 12 out of 18 models (9 subjects described two flat process models) subjects ignored one or more existence constraints. Table 5 shows the number of possible mentions of existence constraints per process model and the number of existence constraints that were ignored by the subjects. Summing up, subjects left out 34 of 78 existence constraints in flat process models.

| Number of | Type | Proc. 1 | Proc. 2 | Proc. 3 | Proc. 4 |
|--|-----------|---------|---------|---------|---------|
| possible mentions of existence constraints | flat | 25 | 8 | 5 | 40 |
| | hierarchy | 28 | 20 | 4 | 65 |
| not mentioned existence constraints | flat | 9 | 1 | 3 | 21 |
| | hierarchy | 19 | 2 | 1 | 30 |

Table 5. Existence constraints

Hierarchical Declarative Process Models. Regarding hierarchical process models, subjects tended to miss less activities. Two out of four subjects forgot to mention one activity in the first process model (11 activities). Regarding the second and third process model, no activities were left out. Three out of five subjects missed one activity in the hierarchical version of the fourth process model (23 activities). In summary, 5 out of 331 activities were missed in hierarchical process models.

Concerning the existence constraints in hierarchical process models, for 11 out of 18 models (9 subjects described two hierarchical process models), one or more existence constraints were not mentioned. As shown in Table 5, 52 from 117 existence constraints were ignored in hierarchical process models.

Flat and Hierarchical Declarative Process Models. As far as the interpretation of constraints is concerned, subjects had relatively little problems irrespective of whether the models were flat or hierarchical. As illustrated in Table 3, 12 different constraint types were used in the experimental material. To accomplish their task, subjects had cheat sheets available and could look constraints they did not know up. Except for the precedence constraint, which caused considerable difficulties, subjects faced no notable problems. Four out of nine subjects used the precedence constraint in a wrong way. According to

Sect. 2, the definition of this constraint is that “*B can only be executed, if A has been executed before*”. The subjects used it the other way round, i.e., “*So if we perform receive incoming good [A] then do quality check [B] should be performed afterwards. . .*” One subject stated the absence of the precedence constraint between two components of a model: “*But still I don’t get the relation between this part and the other one, so this is the problem, because I understand the flow, but I don’t understand the relation between the two parts. Because there is no precedence.*” Additionally, the missing direction of the coexistence constraint caused one subject troubles: “*Let’s see, so I would say you kinda start with these two activities, I’m not sure which one. . .*”

5.3 Combination of Constraints

Constraints between two Activities. The first process model contained two and the fourth process model five situations where two constraints link two activities. In 6 out of these 7 cases, the direction of the constraint arrows are directly opposed to each other. For example, one needs to get offers for interior of an apartment before buying them (precedence constraint). After the interior is bought, it is not reasonable to get new offers (negation response). The subjects had no troubles to understand these situations. However, in the first process model there is a case where a precedence constraint and a chained response constraint link the two activities “*write test*” and “*run tests*”. Both arrows are pointing to the second activity (cf. Fig. 4). The precedence constraint ensures that before the first execution of “*run tests*”, “*write test*” must be executed at least once, i.e., it is not possible to run a test before it was written. The chained response constraint tells us that “*If A has been executed, B must be executed immediately afterwards.*”, meaning that after the test was written, it must be run directly afterwards; 4 out of 9 subjects had troubles with “the second arrow”, i.e., the precedence constraint. Two of them claimed that it is redundant (“*This part is redundant, right?*”), two even thought it is wrong (“*Over this relation, this is a precedence, so I think this is, ah, this can be removed.*”). The other 5 subjects ignored the precedence constraint.

Hidden Dependencies. Three out of the four process models contain hidden dependencies (cf. Sect. 2). Since these interactions are not *explicitly* visible, it is not sufficient that the analyst only relies on the information displayed explicitly, but must carefully examine the process model for these hidden dependencies as well. Our results show that the subjects mostly ignored hidden dependencies, i.e., only in 8 out of 36 models, a hidden dependency was mentioned or found: “*I have to execute prepare lesson in detail at least once, therefore, to fulfill the precedence constraint, I must execute prepare teaching sequence too.*”

6 Discussion

Reading Declarative Process Models. Subjects preferred describing process models in an iterative and sequential way. They started with the entry point of

a component describing it in a sequential way and repeating this procedure for every component of the process model. The sequential way of describing models is surprising, as it is known that declarative process models rather convey circumstantial information (overall conditions that produce an outcome) than sequential information (how the outcome is achieved) [19]. In other words, in an imperative model, sequences are made explicit, e.g., through sequence flows. In a declarative process model, however, such information might not be available at all. As subjects tend to talk about declarative models in a sequential manner, it appears as if they prefer this kind of information. Interestingly, similar observations could be made in a case study on declarative process modeling [17]. Therein, sequential information, such as “*A before B*” or “*then C*” was preferred for communication.

Single Building Blocks. Regarding the interpretation of single building blocks, subjects mentioned activities and constraints when trying to understand the model. Overall, they had relatively little problems with the interpretation of single building blocks. Exceptions seem to be precedence and existence constraints. As a possible explanation these constraints are too simple and are thus not mentioned at all; further, cheat sheets are not used (cf. dual-process theory [20] describing the interplay of implicit unconscious and explicit controlled processes). Another explanation is that subjects were biased by previous knowledge about imperative models. Regarding the precedence constraint, it nearly looks like the arrow used in imperative process modeling notations.

Combining Constraints. The interplay of constraints seems to pose a challenge, especially hidden dependencies. One explanation could be that subjects simply forgot looking for them, as reading declarative models can quickly become too complex for humans to deal with [6]. As mentioned earlier, in 8 out of 36 models subjects found a hidden dependency. In 5 of these 8 cases, they were found in the second process model, which has the smallest number of activities, constraints and constraint types (cf. Table 2). This indicates that if a model is not too complex, subjects will be able to find hidden dependencies. Given this finding, it seems plausible that the *automated* interpretation of constraints can lead to significant improvements regarding the understandability of declarative process models [21, 5].

Differences between Flat and Hierarchical Process Models. Subjects did not distinguish between flat and hierarchical process models when reading the models. They used the same description strategy for components and sub-processes. Interestingly, subjects left out more activities in flat than in hierarchical process models (cf. Sect. 5.2). A reason for this phenomenon could be *abstraction* [22], i.e., hierarchy allows aggregating model information by hiding the internals of a sub-process using a complex activity. Thereby, information can be easier perceived. All other aspects we found could be observed in flat and hierarchical models equally.

Limitations. Our work has the following limitations. First, the number of subjects in the exploratory study is relatively low (9 subjects), hampering result generalization. Nevertheless, it is noteworthy that the sample size is not unusual for this kind of empirical investigation due to the substantial effort to be invested per subject [23]. Second, even though process models used in this study vary in the number of activities, number of constraints, and existence of sub-processes, it remains unclear whether results are applicable to declarative process models in general, e.g., more complex models. Third, all participating subjects indicated academic background, limiting result generalization. However, subjects indicated profound background in business process management, hence, we argue that they can be interpreted as proxies for professionals.

7 Related Work

We investigated the sense-making of declarative process models. The understanding of a declarative process model with respect to modularization has been investigated in [13]. However, opposed to our work, theory rather than empirical data is used for analysis. The role of understanding declarative process models during *modeling* has been investigated in [9]. Similar to our work, it has been postulated that declarative models are most beneficial when sequential information is directly available, as empirically validated in [17, 5]. With respect to the understanding of process models in general, work dealing with the understandability of *imperative* business process models is related. The Guidelines of Modeling (GoM) describe various quality considerations for process models [24]. The so-called ‘Seven Process Modeling Guidelines’ (7PMG) accumulate the insights from various empirical studies, e.g., [25], to develop a set of actions a system analyst may want to undertake to avoid issues with respect to understandability [26]. The understandability of imperative process models is investigated empirically in [2]. As example of understandability issues in conceptual systems, [27] investigates if UML analysis diagrams increase system analysts’ understanding of a domain.

The impact of hierarchy on understandability has been studied in various conceptual modeling languages, such as imperative business process models [28], ER diagrams [29], and UML statechart diagrams [30] (an overview is presented in [22]). Still, none of these works deals with the impact of hierarchy on understandability in declarative process models.

While the effectiveness and usability of design guidelines for multiple diagrams were evaluated in [31], there are neither guidelines for designing nor for easily understanding declarative process models.

8 Summary and Outlook

Declarative approaches to business process modeling have recently attracted interest as they provide a high degree of flexibility [6]. However, the increase in flexibility comes at the cost of understandability, and hence might result in maintainability problems of respective process models [6, 32, 33]. The presented

exploratory study investigates how subjects make sense of declarative business process models and provides insights into occurring problems. The results indicate that subjects read declarative process models in a sequential way. While single constraints caused only minor problems with exception of the precedence constraint, the combination of several constraints seems to be more challenging. More specifically, the subjects of this exploratory study mostly failed to identify hidden dependencies caused by combinations of constraints.

Even though the data we collected provided first insights into the process of understanding declarative models, further investigations are needed. Replications utilizing more complex models seem to be appropriate means for additional empirical tests. Although the think-aloud protocols already provide a detailed view on the reasoning processes of an analyst, we plan to employ eye movement analysis for more detailed analysis. The latter allows identifying areas, the analyst is focusing on in combination with insights on the required cognitive effort (similar to process modeling [34]). Based on these insights, we intend to evolve our work toward empirically founded guidelines enabling better understandability of declarative process models.

References

1. Mylopoulos, J.: Information modeling in the time of the revolution. *Information Systems* **23** (1998) 127–155
2. Reijers, H.A., Mendling, J.: A Study into the Factors that Influence the Understandability of Business Process Models. *SMCA* **41** (2011) 449–462
3. Reichert, M., Weber, B.: *Enabling Flexibility in Process-Aware Information Systems: Challenges, Methods, Technologies*. Springer (2012)
4. Hildebrandt, T., R. Mukkamala, T.S.: Nested dynamic condition response graphs. In: *Proc. FSEN '12*. (2012) 343–350
5. Zugal, S., Pinggera, J., Weber, B.: The impact of testcases on the maintainability of declarative process models. In: *Proc. BPMDS '11*. (2011) 163–177
6. Pesic, M.: *Constraint-Based Workflow Management Systems: Shifting Control to Users*. PhD thesis, TU Eindhoven (2008)
7. Barba, I., Weber, B., Valle, C.D., Ramrez, A.J.: *User Recommendations for the Optimized Execution of Business Processes*. DKE (2013)
8. Zugal, S., Pinggera, J., Weber, B.: Assessing process models with cognitive psychology. In: *Proc. EMISA '11*. (2011) 177–182
9. Zugal, S., Pinggera, J., Weber, B.: Toward Enhanced Life-Cycle Support for Declarative Processes. *Journal of Software: Evolution and Process* **24** (2012) 285–302
10. Ericsson, K.A., Simon, H.A.: *Protocol analysis: Verbal reports as data*. MIT Press (1993)
11. Aalst, W., Pesic, M.: Decserflow: Towards a truly declarative service flow languages. *Lecture Notes in Computer Science* **4184** (2006) 1–23
12. Parnas, D.L.: On the Criteria to be Used in Decomposing Systems into Modules. *Communications of the ACM* **15** (1972) 1053–1058
13. Zugal, S., Soffer, P., Pinggera, J., Weber, B.: Expressiveness and Understandability Considerations of Hierarchy in Declarative Business Process Models. In: *Proc. BPMDS '12*. (2012) 167–181
14. Bassey, M.: *Case study research in educational settings. Doing qualitative research in educational settings*. Open University Press (1999)

15. Corbin, J., Strauss, A.: Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory. SAGE Publications (2007)
16. Haisjackl, C.: Test Driven Modeling meets Declarative Process Modeling A Case Study. Master's thesis, University of Innsbruck (2012)
17. Zugal, S., Haisjackl, C., Pinggera, J., Weber, B.: Empirical Evaluation of Test Driven Modeling. IJISMD (to appear, available online)
18. Khatri, V., Vessey, I., Ramesh, P.C.V., Park, S.J.: Understanding Conceptual Schemas: Exploring the Role of Application and IS Domain Knowledge. *Information Systems Research* **17** (2006) 81–99
19. Fahland, D., Mendling, J., Reijers, H.A., Weber, B., Weidlich, M., Zugal, S.: Declarative versus Imperative Process Modeling Languages: The Issue of Understandability. In: *Proc. EMMSAD '09*. (2009) 353–366
20. Kahneman, D.: Maps of bounded rationality: A perspective on intuitive judgment and choice. *Nobel prize lecture* **8** (2002) 449–489
21. Zugal, S., Pinggera, J., Reijers, H., Reichert, M., Weber, B.: Making the Case for Measuring Mental Effort. In: *Proc. EESSMod '12*. (2012) 37–42
22. Zugal, S., Pinggera, J., Mendling, J., Reijers, H., Weber, B.: Assessing the Impact of Hierarchy on Model Understandability-A Cognitive Perspective. In: *Proc. EESSMod '11*. (2011) 123–133
23. Costain, G.F.: Cognitive Support During Object-oriented Software Development: The Case of UML Diagrams. PhD thesis, University of Auckland (2007)
24. Becker, J., Rosemann, M., Uthmann, C.: Guidelines of Business Process Modeling. In: *Business Process Management, Models, Techniques, and Empirical Studies*, London, UK, Springer-Verlag (2000) 30–49
25. Mendling, J., Verbeek, H., van Dongen, B., van der Aalst, W., Neumann, G.: Detection and prediction of errors in eps of the sap reference model. *DKE* **64** (2008) 312–329
26. Mendling, J., Reijers, H.A., van der Aalst, W.M.P.: Seven process modeling guidelines (7pmg). *Information & Software Technology* **52** (2010) 127–136
27. Burton-Jones, A., N.Meso, P.: Conceptualizing systems for understanding: An empirical test of decomposition principles in object-oriented analysis. *Information Systems Research* **17** (2006) 38–60
28. Reijers, H., Mendling, J., Dijkman, R.: Human and automatic modularizations of process models to enhance their comprehension. *Inf. Systems* **36** (2011) 881–897
29. Moody, D.L.: Cognitive Load Effects on End User Understanding of Conceptual Models: An Experimental Analysis. In: *Proc. ADBIS '04*. (2004) 129–143
30. Cruz-Lemus, J.A., Genero, M., Morasca, S., Piattini, M.: Using Practitioners for Assessing the Understandability of UML Statechart Diagrams with Composite States. In: *Proc. ER Workshops '07*. (2007) 213–222
31. Kim, J., Hahn, J., Hahn, H.: How do we understand a system with (so) many diagrams? cognitive integration processes in diagrammatic reasoning. *Information Systems Research* **11** (2000) 284–303
32. Weber, B., Reijers, H.A., Zugal, S., Wild, W.: The Declarative Approach to Business Process Execution: An Empirical Test. In: *Proc. CAiSE '09*. (2009) 270–285
33. Zugal, S., Pinggera, J., Weber, B.: Toward Enhanced Life-Cycle Support for Declarative Processes. *Journal of Software: Evolution and Process* **24** (2012) 285–302
34. Pinggera, J., Furtner, M., Martini, M., Sachse, P., Reiter, K., Zugal, S., Weber, B.: Investigating the Process of Process Modeling with Eye Movement Analysis. In: *Proc. ER-BPM '12*. (2013) 438–450